wCorr Arguments

Paul Bailey

2016-03-04

This vignette explores two Boolean switches in the wCorr package. First, the `ML` switch allows for either a non-MLE (but consistent) esitimate of the nusiance parameters that define the binning process to be used (`ML=FALSE`) or for the nusiance parameters to be estimated using the MLE (`ML=TRUE`). Second the `fast` argument gives the option to use a pure R implementation (`fast=FALSE`) or an implementation that relies on the `Rcpp` and `RcppArmadillo` packages (`fast=TRUE`).

Numerical simulations in this vignette show that differences in the results are essentially unaffected by either of these switches.

The *wCorr Formulas* vignette describes the statistical properties of the correlation estimators in the package and has a more complete derivation of the likelihood functions.

The `ML` switch

The correlation coefficients between two vectors of random variables that are jointly bivariate normal–call the vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$.

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N \left[\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \boldsymbol{\Sigma} \right]$$

where $N(\mathbf{A}, \boldsymbol{\Sigma})$ is the bivariate normal distribution with mean $\boldsymbol{A}$ and covariance $\boldsymbol{\Sigma}$.

Polyserial computation

the likelihood function for an individual observation of the polyserial is[1]

$$\Pr\left(\rho = r, \boldsymbol{\theta}; Z = z_i, M = m_i\right) = \phi(z_i) \left[\Phi\left(\frac{\theta_{m_i+2} - r \cdot z_i}{\sqrt{1-r^2}} \right) - \Phi\left(\frac{\theta_{m_i+1} - r \cdot z_i}{\sqrt{1-r^2}} \right) \right]$$

where ρ is the correlation between $\boldsymbol{X}$ and $\boldsymbol{Y}$, $\boldsymbol{Z}$ is the normalized version of $\boldsymbol{X}$, and $\boldsymbol{M}$ is a discretized version of $\boldsymbol{Y}$, using $\boldsymbol{\theta}$ as cut points as described in the "*Corr Formulas* vignette.

The log-likelihood is then

$$\ell(\rho, \boldsymbol{\theta}; z, m) = \sum_i w_i \ln\left[\Pr\left(\rho = r, \boldsymbol{\theta}; Z = z_i, M = m_i\right) \right]$$

The derivatives of ℓ can be computed but are not readily computed and so when the `ML` argumet is set to `FALSE` (the default) a one dimensional optimization of ρ is calculated using `stats::optimize`. When the `ML` argument is set to `TRUE` a multi-dimensional optimization is done for ρ and $\boldsymbol{\theta}$ using `minqa::bobyqa`.

[1] See the "wCorr Formulas" vignette for a more complete description and motivation for the polyserial correlations's likelihood function.

Polychoric computation

the likelihood function for the polychoric is[2]

$$\Pr\left(\rho = r, \boldsymbol{\theta}, \boldsymbol{\theta}'; P = p_i, M = m_i\right) = \int_{\theta'_{p_i+1}}^{\theta'_{p_i+2}} dx \int_{\theta_{m_i+1}}^{\theta_{m_i+2}} dy f(x, y|\rho = r)$$

where $f(x, y|r)$ is the noramlzied bivariate normal distribution with correlation ρ.

The log-likelihood is then

$$\ell(\rho, \boldsymbol{\theta}, \boldsymbol{\theta}'; p, m) = \sum_i w_i \ln\left[\Pr\left(\rho = r, \boldsymbol{\theta}, \boldsymbol{\theta}'; P = p_i, M = m_i\right)\right]$$

The derivatives of ℓ can be computed but are not readily computed and so when the `ML` argumet is set to `FALSE` (the default) a one dimensional optimization of ρ is calculated using `stats::optimize`. When the `ML` argument is set to `TRUE` a multi-dimensional optimization is done for ρ, $\boldsymbol{\theta}$, and $\boldsymbol{\theta}'$ using `minqa::bobyqa`.

General setup for the unweighted case

A simulation is run several times. For each itteration, the following procedure is used:

- select the number of observations (n)
- select a true correlation coefficient ρ
- generate $\boldsymbol{X}$ and $\boldsymbol{Y}$ to be bivariate normally distributed using a pseudo-Random Number Generator (RNG)
- using a pseudo-RNG, select the the number of bins for $\boldsymbol{M}$ and $\boldsymbol{P}$ (t and t') independantly from the set {2, 3, 4, 5}
- select the bin boundaries for $\boldsymbol{M}$ and $\boldsymbol{P}$ ($\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$) by sorting the results of t and t' draws, respectively, from a normal distribution using a pseudo-RNG
- confirm that at least 2 levels of each of $\boldsymbol{M}$ and $\boldsymbol{P}$ are occupied (if not, retrun to generating $\boldsymbol{X}$ and $\boldsymbol{Y}$)
- calculate and record relevant statistics

when the exact method of selecting a parameter (such as n) is not noted in the above description it is described as part of each simulation.

ML switch

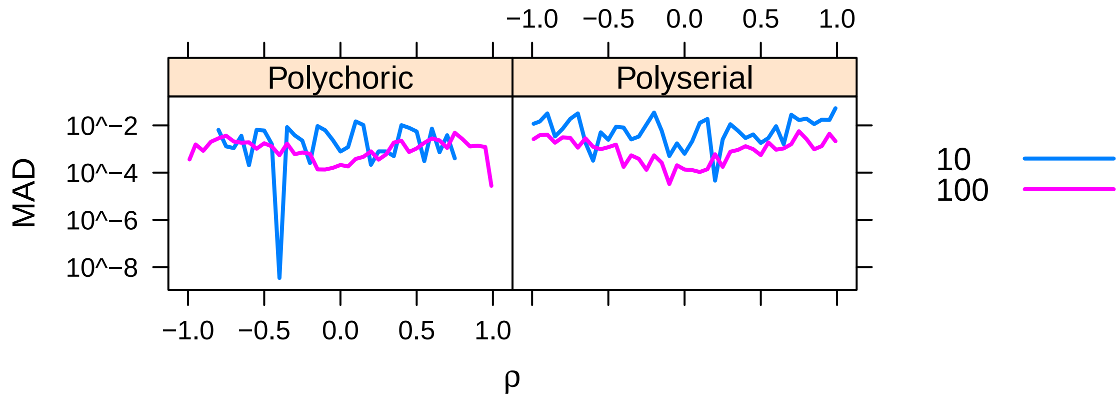
A simulation was done at each level of the cartesian product of $\texttt{ML} \in \{\texttt{TRUE}, \texttt{FALSE}\}$, $\rho \in (-0.99, -0.95, -0.90, -0.85, ..., 0.95, 0.99$ and $n \in \{10, 100, 1000\}$. For precision, each iteration is run three times. The compulation is run so that the same values of the variables are used for `ML=TRUE` as `ML=FALSE` and then the statistics are comared between the two sets of results. where MAD is the mean absolute difference and is given by

$$MAD = |r_{ML=TRUE} - r_{ML=FALSE}|$$

where $r_{ML=TRUE}$ is the estimated correlation when `ML=TRUE` and $r_{ML=FALSE}$ is the estimated correlation when `ML=FALSE`.

[2] See the "wCorr Formulas" vignette for a more complete description and motivation for the polychoric correlations's likelihood function.

This is a plot of the MAD as a function of the true correlation coefficnet.



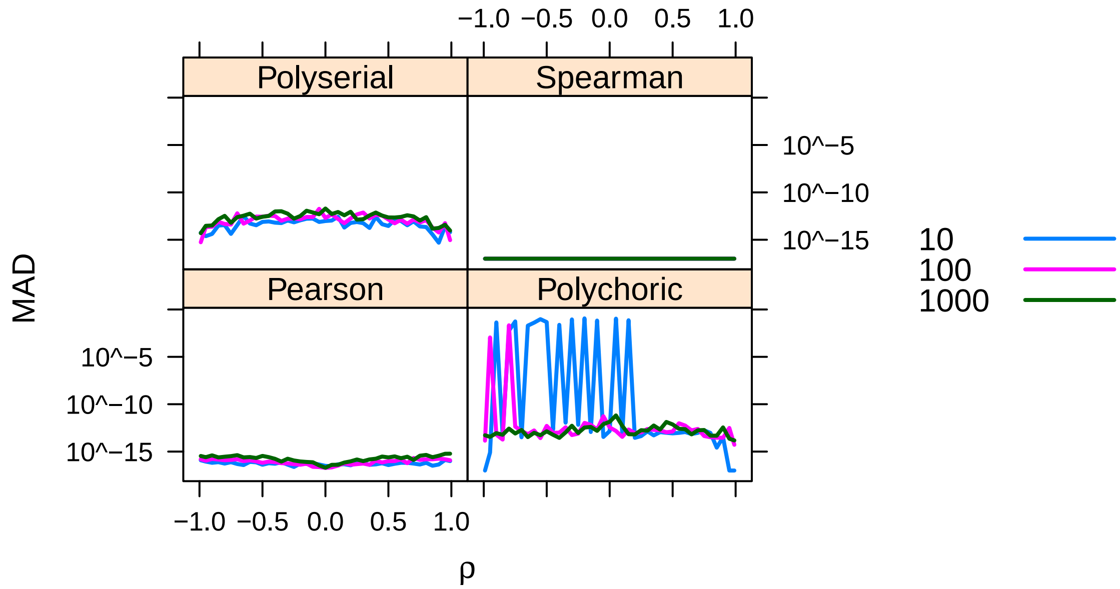
This table shows the MAD by n and correlation type.

Correlation type	n	MAD
Polychoric	10	0.0032023
Polychoric	100	0.0013196
Polyserial	10	0.0108751
Polyserial	100	0.0014155

fast switch

This section looks at the agreement between the pure R implementation of the optimizations and the `Rcpp` and `RcppArmadillo` impelemntation. The code can compute with either option by setting `fast=FALSE` (pure R) or `fast=TRUE` (Rcpp).

This is the summary of all differences between the `fast=TRUE` and `fast=FALSE runs` for the polyserial



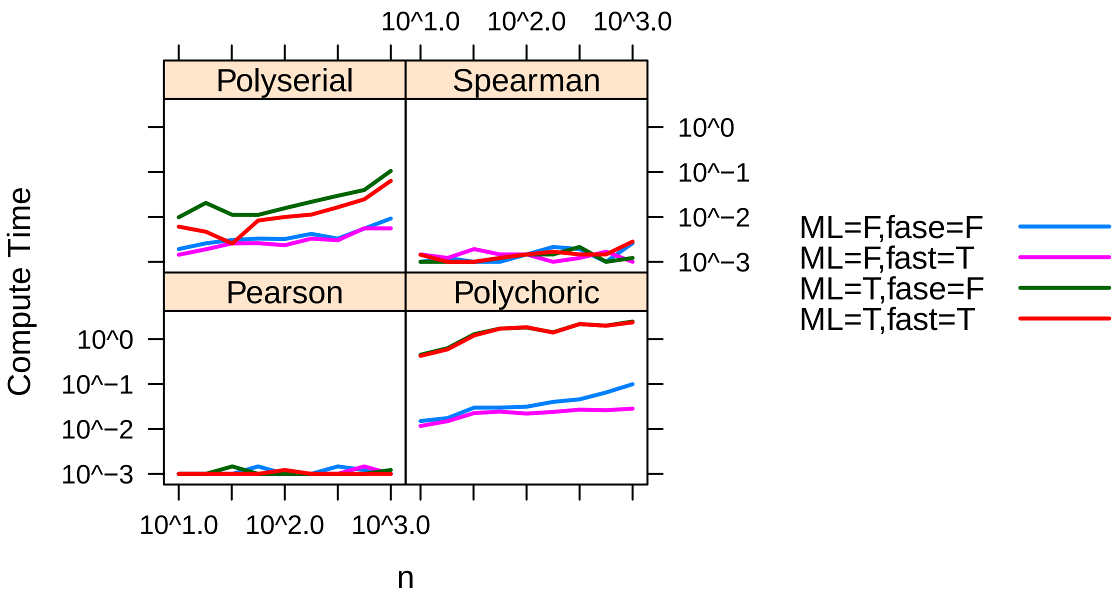
This table shows the MAD by n and correlation type.

Correlation type	n	MAD
Pearson	10	0.0000000
Pearson	100	0.0000000
Pearson	1000	0.0000000
Polychoric	10	0.0190487
Polychoric	100	0.0005456
Polychoric	1000	0.0000000
Polyserial	10	0.0000000
Polyserial	100	0.0000000
Polyserial	1000	0.0000000
Spearman	10	0.0000000
Spearman	100	0.0000000
Spearman	1000	0.0000000

Implications for speed

A simulation was done at each level of the cartesian product of `ML` $\in \{$`TRUE`, `FALSE`$\}$, `fast` $\in \{$`TRUE`, `FALSE`$\}$, $\rho \in (-0.99, -0.95, -0.90, -0.85, ..., 0.95, 0.99)$, and $n \in \{10^1, 10^{1.25}, 10^{1.5}, ..., 10^7\}$. For precision, each iteration is run three times. The compulation is run so that the same values of the variables are used all four levels of `ML` and `fast`. The variety of correlations is chosen so that the results represent an average of possible values of ρ.

The following plot shows the mean compute time versus n.


Polyserial
Spearman
Pearson
Polychoric
Compute Time
n
10^1.0
10^2.0
10^3.0
10^0
10^−1
10^−2
10^−3
ML=F,fase=F
ML=F,fast=T
ML=T,fase=F
ML=T,fast=T